Oblikovanje i analiza algoritama

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

26. rujna, 2023.

Kontakt

- ured 226, drugi kat
- e-mail: matmih@math.hr
- konzultacije: ponedjeljak, 8-10h (uz prethodnu najavu mailom)

Sadržaj kolegija

- Uvod u složenost
  - Pojam i svojstva algoritma
  - Što je složenost
  - Relacije asimptotskog ponašanja funkcija
- Rekurzivni algoritmi
- Analiza složenosti nekih (vama) poznatih algoritama
  - Sortiranje - Quicksort
  - Primov i Kruskalov algoritam
- Metode za oblikovanje efikasnih algoritama
  - Podijeli pa vladaj (eng. *Divide and Conquer*)
  - Pohlepni (eng. *Greedy*) pristup
  - Dinamičko programiranje (eng. *Dynamic Programming*)
  - ...
- Teško rješivi problemi
  - Klase P i NP
  - Primjeri $\mathcal{NP}$-potpunih i $\mathcal{NP}$-teških problema
  - Rješavanje takvih problema

Literatura

- M. Mihelčić, Nastavni materijali , PMF, Matematički odsjek, 2023. (`https://www.pmf.unizg.hr/math/predmet/oaa`). Bazirano na predavanjima prof. Saše Singera i prof. Goranke Nogo.
- S. Singer, Nastavni materijali , PMF, Matematički odsjek (`https://web.math.pmf.unizg.hr/nastava/oaa/`)
- J. Kleinberg , E. Tardos , Algorithm Design , Pearson Education , Inc., 2005.
- M. T. Goodrich , R. Tamassia , Algorithm Design and Applications , Wiley , 2015.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, Fourth edition, The MIT Press, 2022.

Način polaganja kolegija

- Pohađanje nastave: minimalno 70% predavanja/vježbi/seminara kao preduvjet izlaska na ispit.
- Projektni zadatak (30% ocjene) - održava se **samo jednom**.
- Ispit (70% ocjene) - 2 ispitna roka (4 ispitna termina).

Tablica ocjenjivanja:

45 - 59: 2

60 - 74: 3

75 - 89: 4

90 ili više: 5

Uvod u složenost

Neformalan opis algoritma:

Algoritam je metoda, proces, postupak, tehnika ili pravilo za rješavanje neke klase problema (obično računalom).

Složenost algoritma je cijena korištenja (izvođenja) tog algoritma za rješavanje jednog problema iz klase. Obično se mjeri u vremenu izvršavanja, potrebnoj memoriji ili sličnim relevantnim pojmovima (ubrzanje - eng. *speedup*, učinkovitost - eng. *efficiency*, paralelizam - eng. *parallelism*, labavost - eng. *slackness*) kod paralelnih algoritama).

Primjeri:

- Množenje prirodnih brojeva
- Euklidov algoritam
- Binarno pretraživanje
- Upute za sastavljanje/korištenje nekog uređaja.

Uvod u složenost

Algoritam se sastoji od **konačnog** niza koraka instrukcija ili naredbi.

Izvođenje tog niza naredbi mora biti objektivan proces koji se, u načelu, mora moći reproducirati. Također, izvođenje ne smije zahtijevati neku intuiciju ili kreativnost.

Napomena: Postoji važna klasa algoritama tzv . probabilistički algoritmi, kod kojih je reproducibilnost znatno teže postići (iako ne nemoguće - uz adekvatnu implementaciju).

Zahtjevi na niz koraka:

- Postoji jednoznačno određeni početni korak.
- Po završetku nekog koraka, točno je određen sljedeći korak koji treba obaviti (determiniranost algoritma)
- Postupak uvijek završava u konačno mnogo koraka, za bilo koji problem iz neke klase.
- Postupak se sastoji od konačnog broja osnovnih (elementarnih, primitivnih) instrukcija. Značenje tih osnovnih instrukcija je jednoznačno definirano za izvršitelja postupka (stroj ili osoba).

Uvod u složenost

Svojstva algoritma:
a) ulaz, b) izlaz, c) konačnost, d) definiranost i nedvosmislenost, e) efikasnost.

Ulaz:

- Svaki algoritam ima nula ili više, ali konačno mnogo, vrijednosti koje treba zadati na početku, prije nego što počne izvršavanje algoritma.
- Ulazne vrijednosti definiraju konkretan primjerak zadaću (instance) problema kojega treba riješiti.
- Svaka ulazna vrijednost uzima se iz nekog unaprijed zadanog skupa dozvoljenih objekata.
- Kartezijev produkt tih skupova je domena ili područje definicije problema.
- Pretpostavljamo da je ulaz korektno zadan. Programska realizacija algoritma može imati dodatne provjere korektnosti ulaza.

Uvod u složenost

Napomena: Algoritmi koje ćemo promatrati, u principu rade za bilo koji ulaz. Npr . algoritmi na prirodnim brojevima rade za sve prirodne brojeve. No, računala mogu reprezentirati samo konačni podskup skupa prirodnih brojeva.

Izlaz:

- Svaki algoritam mora imati barem jedan izlaz, jer inače nije ostavio trag svog izvršavanja.
- To je traženo „rješenje" našeg problema.

Konačnost:

- Svaki algoritam, za svaki ulaz, mora završiti u konačno mnogo koraka.

Definiranost i nedvosmislenost:

- Svaka osnovna instrukcija mora biti jednoznačno definirana (barem za izvršitelja).
- Što su osnovne instrukcije je pitanje dogovora.

Uvod u složenost

Efikasnost:

- Izvršavanje algoritma ne smije trajati predugo.
- Što je „predugo" ovisi o problemu i potrebi.
- Postoji klasa problema za koje efikasno rješenje postoji samo za male zadaće.

Primjer problema za koji ne postoji efikasno rješenje:
ispis svih permutacija skupa $\{1, \ldots, n\}$.

Za $n = 100$ imamo:

$$100! = 933262154439441526816992388562667004907159682643$$
$$816214685929638952175999932299156089414639761565$$
$$182862536979208272237582511852109168640000000$$
$$0000000000000000 \approx 9.33262 \cdot 10^{157}$$

Uvod u složenost

Efikasnost:

- Rješavanje problema (izračunavanje u širem smislu) troši vrijeme.
- Neki problemi zahtijevaju vrlo mnogo vremena, a neki ne.
- Za određeni problem može postojati više algoritama za rješenje. Kojega ćemo upotrijebiti? Najboljeg. Ali po kojim kriterijima?
- Predmet istraživanja - količina resursa izračunavanja koji su potrebni za određene vrste izračunavanja.

Količina potrebnih resursa: **složenost algoritma**.

Uvod u složenost

Što možemo mjeriti? Što nas najčešće zanima?

Neke mjere:

- prostorna složenost (eng. *space complexity*)
- vremenska složenost (eng. *time complexity*)
- aritmetička složenost (eng. *arithmetic complexity*)
- komunikacijska složenost (eng. *communication complexity*)

Koja je mjera važnija? Odgovor nije uvijek jednoznačan.
Može se pokazati da je prostorna složenost bitno manja od aritmetičke, odnosno vremenske složenosti.
Prostornu složenost obično nećemo navoditi.

Uvod u složenost

Na kolegiju će nam biti zanimljivije **aritmetička** i **vremenska** složenost.
Aritmetička složenost gotovo ne ovisi o arhitekturi računala, za razliku od vremenske složenosti.
Na sekvencijalnim arhitekturama, te dvije složenosti su **sličnog reda veličine**, dok paralelno izvođenje aritmetičkih operacija **može bitno smanjiti vremensku složenost**.

Uvod u složenost

O čemu ovisi složenost?
Pretpostavimo da imamo dva algoritma koji rješavaju isti problem, tj. imaju isti skup zadaća koje mogu riješiti.
Kako ćemo ih usporediti? Trebamo izmjeriti složenost tih algoritama za svaku zadaću tog problema.
Složenost:

- zadaća $\rightarrow$ broj
- problem $\rightarrow \mathbb{R}^+$(ili $\mathbb{N}$)

Kako efikasno riješiti sve zadaće problema?
Nemoguća misija!

Uvod u složenost

Složenost - funkcija veličine zadaće koju treba riješiti
Veličina zadaće, u oznaci, x je:

- broj bitova potrebnih za reprezentaciju zadaće x na ulazu algoritma, uz korištenje jednoznačno definiranog i razumno kompaktnog načina kodiranja.

Može se pokazati da su svi načini razumnog kodiranja međusobno ekvivalentni u smislu da su gotovo proporcionalni, tj . ako je $|x_1|$ duljina zadaće x u kodu 1, a $|x_2|$ duljina zadaće x u kodu 2, onda postoje konstante $c_1,\ c_2 > 0$, takve da za svaku zadaću x vrijedi:

$$c_1|x_1| \leq |x_2| \leq c_2|x_1|$$

Uvod u složenost

Uočimo da postoji više zadaća s istom veličinom.
Dva pitanja:

- po kojem principu pridružujemo broj zadaćama iz iste klase s istom veličinom
- kako stvarno dolazimo do tog broja (što brojimo)?

Npr. možemo brojati:

- izvršene aritmetičke (floating point) operacije
- napravljene usporedbe
- prolaske kroz petlju

Uvod u složenost

Uočimo da postoji više zadaća s istom veličinom.
Dva pitanja:

- po kojem principu pridružujemo broj zadaćama iz iste klase s istom veličinom
- kako stvarno dolazimo do tog broja (što brojimo)?

Npr. možemo brojati:

- izvršene aritmetičke (floating point) operacije
- napravljene usporedbe
- prolaske kroz petlju

Uvod u složenost

Postoje dva osnovna pristupa u analizi algoritama maksimalni i prosječni.
Najgora složenost (emg. *worst case complexity*)

- Za danu veličinu problema uzimamo najveće moguće vrijeme (ili nešto drugo), za sve zadaće te veličine.
- To je najgori mogući scenarij.
- Najčešće se koristi jer se najlakše provodi.

Prosječna složenost (average complexity)

- Tražimo prosječno trajanje (ili nešto drugo), po svim zadaćama iste veličine.
- To je mnogo realističnija i korisnija mjera.
- Često puno teža za izračunati.

Uvod u složenost

Kako nalazimo složenost (onu koju želimo najgoru ili prosječnu)?
Metodološki gledano, imamo dva osnovna pristupa.

- empirijski (a posteriori, naknadni)
- teorijski (a priori, prethodni)

Kombinacija prethodna dva pristupa daje treći hibridni.
Empirijski pristup

- Programiramo algoritam koji želimo analizirati u nekom jeziku, na nekom računalu.
- Taj program izvodimo na nekom uzorku zadaća i mjerimo, potrebne resurse (vrijeme).
- Zvuči jednostavno. No, je li baš tako?

Uvod u složenost

Kako doći do oblika funkcije?

- Eksperimentiranjem izaberemo nekoliko funkcija i gledamo koja od njih daje bolje rezultate.
- A priori analizom, tj. teorijom.

Druga metoda je očito povoljnija. Tako dobivamo hibridni pristup.
Teorijski pristup

- Matematičkom analizom algoritma određujemo funkciju složenosti u ovisnosti o veličini zadaće.
- Da to možemo napraviti, moramo izabrati neki model izvršavanja algoritma i dogovoriti se o složenosti osnovnih operacija u tom modelu.
- To nije uvijek jednostavno, pogotovo ako želimo da model bude realan.
- Problem modela, izbora i složenosti elementarnih operacija možemo smatrati manom teorijskog pristupa.

Uvod u složenost

Prednosti teorijskog pristupa

- Ne ovisi o računalu koje koristimo, programskom jeziku u kojem pišemo ili čak o vještini programera.
- Štedimo vrijeme programera i stroja za programiranje i izvođenje nekog, potencijalno, neefikasnog algoritma.
- Dopušta nam analizu efikasnosti algoritma na zadaćama bilo koje veličine.
- Teorijski pristup daje oblik funkcije složenosti u ovisnosti o veličini zadaće.

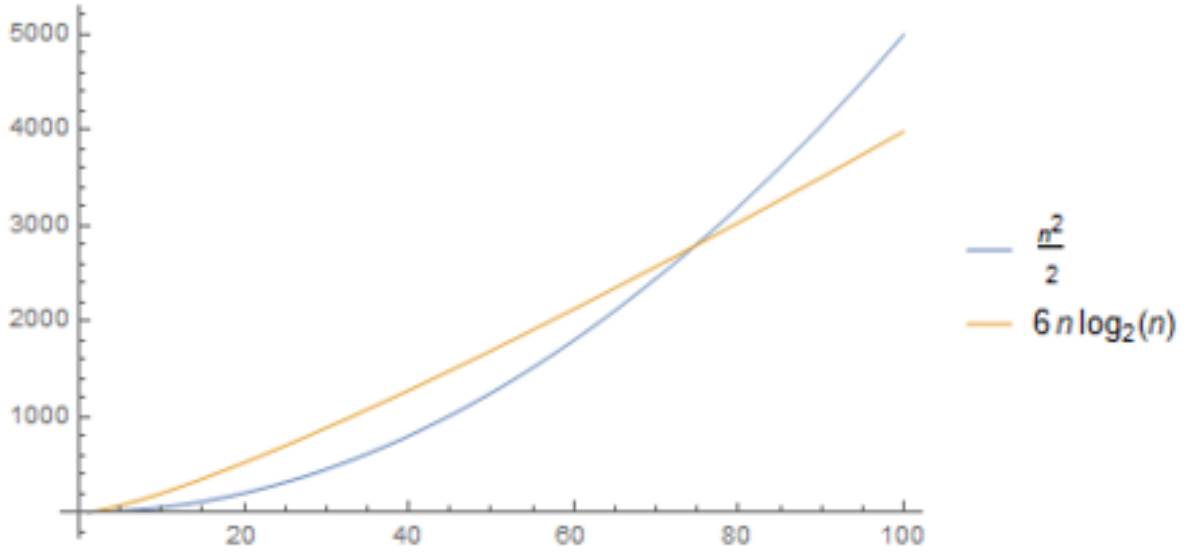
Uvod u složenost

U teorijskom pristupu ostaje još jedan problem.

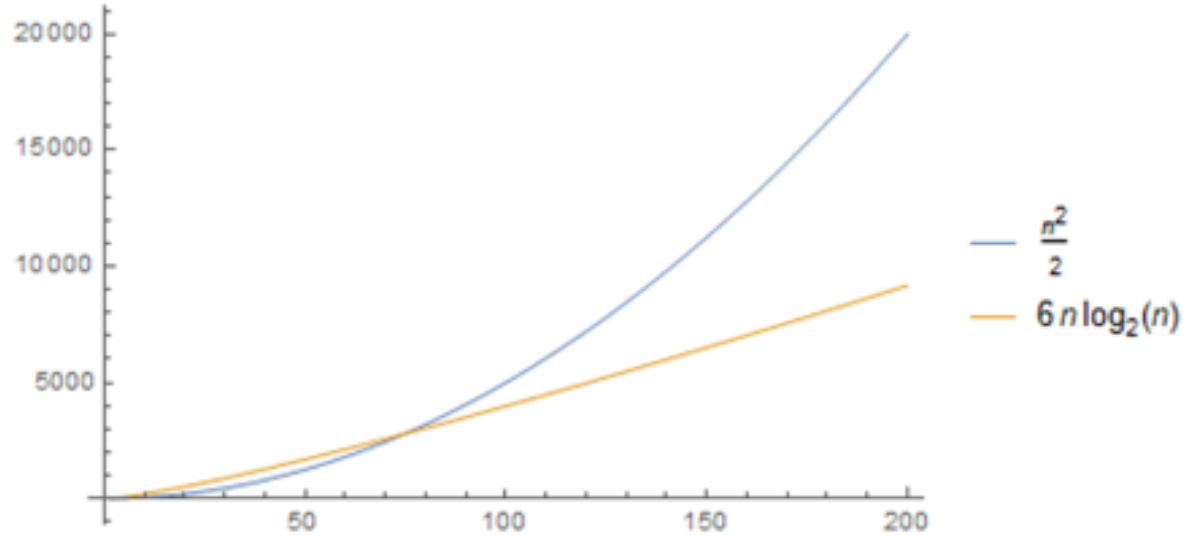
- U kojim jedinicama ćemo izražavati (vremensku) složenost algoritma?
- Sekunde (ili slična jedinica) očito ne dolaze u obzir!
- Treba nam „standardno" računalo kao referentna točka za baždarenje svih mjerenja.
- Možemo odabrati jedan od standardnih modela izračunavanja, npr. Turingov stroj i sve mjeriti brojem osnovnih operacija tog stroja.
- To je nepraktično za većinu naših potreba.
- Nama treba jednostavni, pa makar i pomalo neprecizni način za usporedbu algoritama.
- Promatrat ćemo ponašanje algoritma za velike zadaće!
- Funkcije složenosti algoritama mogu biti vrlo komplicirane.
- Dakle, trebamo uvesti pojmove koji omogućavaju usporedbu funkcija za velike argumente.

Uvod u složenost

```
Plot[{n^2/2, 6 n Log[2, n]}, {n, 1, 100}, AxesOrigin → {0, 0}, PlotLegends → "Expressions"]
```

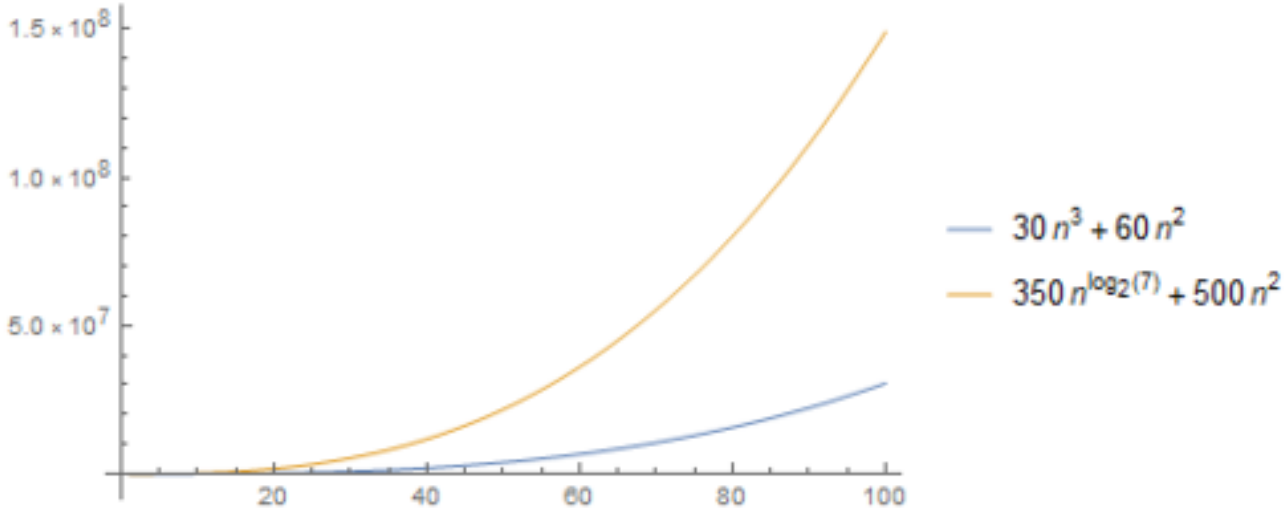



```
Plot[{n^2/2, 6 n Log[2, n]}, {n, 1, 200}, AxesOrigin → {0, 0}, PlotLegends → "Expressions"]
```

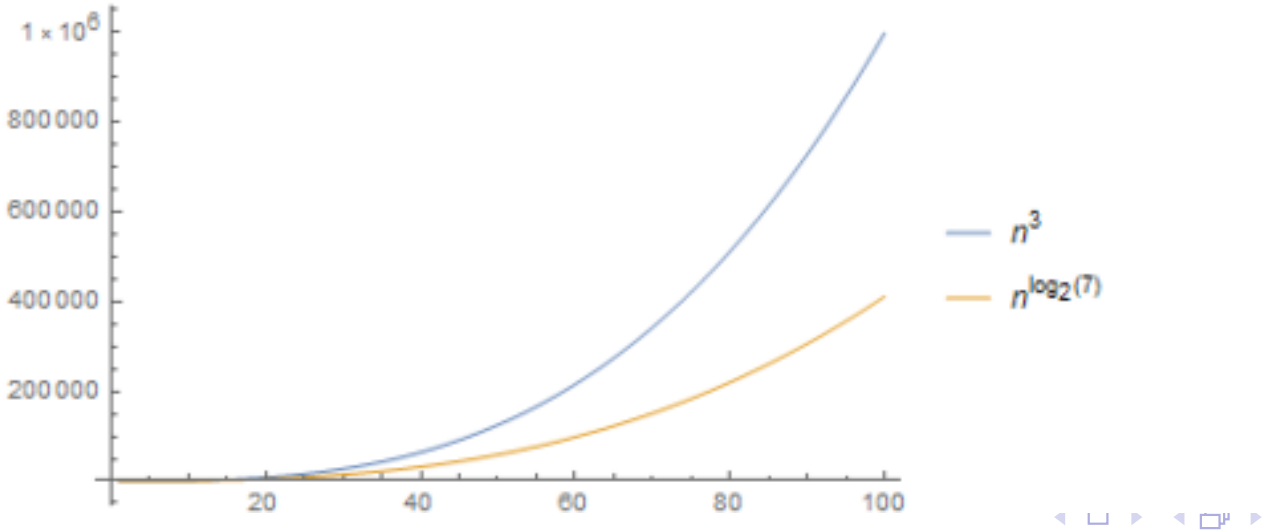


Uvod u složenost

```
Plot[{30 n^3 + 60 n^2, 350 n^Log[2, 7] + 500 n^2}, {n, 1, 100}, AxesOrigin → {0, 0},
 PlotLegends → "Expressions"]
```




```
Plot[{n^3, n^Log[2, 7]}, {n, 1, 100}, AxesOrigin → {0, 0}, PlotLegends → "Expressions"]
```



Uvod u složenost

Za velike zadaće zanimat će nas samo onaj dio te funkcije koji dominantno utječe na njeno ponašanje za velike argumente.
Prirodno je pretpostaviti da za različite implementacije istog algoritma vrijedi sljedeći **princip invarijantnosti**:

- Različite implementacije istog algoritma, razlikuju se u efikasnosti najviše do na multiplikativnu konstantu.

Relacije asimptotskog ponašanja funkcija:

Definicija 1. Neka su $f,\ g : D \mapsto \mathbb{R}$ realne funkcije na odozgo neograničenom podskupu $D \subseteq \mathbb{R}$.

Veliko O

- f **nije većeg reda veličine** od g, ili f **ne raste brže** od g, u oznaci

$$f(x) \in \mathcal{O}(g(x)),\ (x \to \infty)$$

ako postoje realna konstanta $C > 0$ i $x_0 \in D$, takvi da $\forall x > x_0$ $|f(x)| < C|g(x)|$.

Primjeri:

- $\frac{x^2}{2} \in \mathcal{O}(x^2)$
- $6x \cdot log_2 x + 6x \in \mathcal{O}(x \cdot log_2 x)$
- $6x \cdot log_2 x + 6x \in \mathcal{O}(x^2)$

Za koje C i x_0 vrijede tvrdnje?

Malo o

- f je **manjeg reda veličine** od g ili f **raste sporije** od g, u oznaci

$$f(x) \in o(g(x)), \ (x \to \infty)$$

ako postoji $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ i jednak je 0.

Primjeri:

- $x^{k-1} \in \mathcal{O}(x^k)$
- $sinx \in \mathcal{O}(x)$
- $\frac{1}{x} \in \mathcal{O}(1)$

Veliko θ

- f je **istog reda veličine** kao g ili f **raste istom brzinom** kao g, u oznaci

$$f(x) \in \theta(g(x)),\ (x \to \infty)$$

ako postoje realne konstante $c_1 > 0$, $c_2 > 0$ i $x_0 \in D$, takvi da je $\forall x > x_0$

$$c_1|g(x)| < |f(x)| < c_2|g(x)|.$$

Primjeri:

- $\frac{x^2}{2} + 3x \in \theta(x^2)$
- $(x+1)^2 \in \theta(x^2)$

Asimptotska jednakost $\sim$

- f i g su **asimptotski jednake**, u oznaci

$$f(x) \sim g(x), \ (x \to \infty)$$

ako postoji

$$\lim_{x \to \infty} \frac{f(x)}{g(x)}$$

i jednak je 1.

Uvod u složenost

Primjeri:

- $\frac{x^2}{2} + 3x \sim \frac{x^2}{2}$ (izdvajamo dominantni član)
- $(3x+1)^4 \sim 81x^4$

Veliko Ω

- f **nije manjeg reda veličine** od g, ili f **raste barem jednako brzo** kao i g, u oznaci

$$f(x) \in \Omega(g(x)),\ (x \to \infty)$$

  ako **nije** $f(x) \in \mathcal{O}(g(x))$

- nije manjeg reda veličine smo definirali kao negaciju od $\mathcal{O}$.

Definicija u afirmativnom obliku:

Uz pretpostavku da je $g(x) \neq 0$ za sve dovoljno velike x, to znači da postoje $\varepsilon > 0$ i niz $x_n \in D,\ n \in \mathbb{N},\ x_n \to \infty$, takvi da $\forall n \in \mathbb{N}$

$$|f(x_n)| > \varepsilon |g(x_n)|$$

Uvod u složenost

Primjeri:

- $\frac{x^2}{2} \in \Omega(x^2)$
- $\frac{x^2}{2} + 3x \in \Omega(x)$
- $\frac{x^2}{2} + 3x \in \Omega(x^2)$

Malo ω

- f **je većeg reda veličine** od g, ili f **raste brže** od g, u oznaci

$$f(x) \in \omega(g(x)),\ (x \to \infty)$$

ako **nije** $f(x) = \mathcal{O}(g(x))$

To znači da $\exists$ nizovi $\varepsilon_n \to \infty,\ x_n \to \infty$ takvi da $\forall n \in \mathbb{N}$

$$|f(x_n)| > \varepsilon_n |g(x_n)|$$

Uvod u složenost

Primjeri:

- $\frac{x^2}{2} \in \omega(x)$
- $\frac{x^2}{2} + 3x \in \omega(x)$
- $x^2 \cdot log_2 x \in \omega(x^2)$

Neka vrijede pretpostavke kao u Definiciji 1.

Definicija 2. Funkcije f i g su asimptotski proporcionalne ako postoji konstanta $c \neq 0$ takva da je:

$$f(x) \sim cg(x)$$

Propozicija 1.

- Relacije asimptotskog ponašanja θ i $\sim$ su relacije ekvivalencije na R^D - skupu svih funkcija $f : D \mapsto \mathbb{R}$.
- Na skupu klasa ekvivalencije R^D/θ, relacije $\mathcal{o}$ i $\mathcal{O}$ su relacije parcijalnog uređaja.

Zašto baš na R^D/θ?

Uvod u složenost

Propozicija 1.

- Relacija Θ je simetrična. Ako je $f(x) \in \Theta(g(x))$, onda je i $g(x) \in \Theta(f(x))$.
- Relacija $\mathcal{O}$ je antisimetrična na R^D/θ, tj. Ako je $f(x) \in \mathcal{O}(g(x))$ i $g(x) \in \mathcal{O}(f(x))$, tada je $f(x) = \Theta(g(x))$ i $g(x) = \Theta(f(x))$.

Zadatak:

- Ako je $f(x) \in \theta(g(x))$ i ako konstante $c_1,\ c_2$ možemo odabrati proizvoljno bliske, tj. $\forall \varepsilon > 0,\ \exists c_1(\varepsilon),\ c_2(\varepsilon) > 0$ i $\exists x_0(\varepsilon) \in D$ takvi da je $c_2(\varepsilon) - c_1(\varepsilon) < \varepsilon$ i

$$c_1(\varepsilon)|g(x)| < |f(x)| < c_2(\varepsilon)|g(x)|,\ \forall x > x_0(\varepsilon)$$

onda su $|f|$ i $|g|$ **asimptotski proporcionalne**, tj. $\exists c > 0$ takav da je $|f(x)| \sim c|g(x)|$.

Definicija 3.

Ako su f i g nenegativne funkcije i ako je $f(x) = \Theta(g(x))$, tada kažemo da su f i g **kodominantne**.

Definicija 4.

Neka je $f : D \mapsto \mathbb{R}_0^+$ nenegativna funkcija na odozgo neograničenom podskupu $D \subseteq \mathbb{R}_0^+$ i pretpostavimo da f neograničeno raste $\lim_{x\to\infty} f(x) = \infty$.

- f je **blagog rasta**, ako f raste sporije od bilo koje pozitivne potencije, tj. $\forall \varepsilon > 0$ je $f(x) \in o(x^\varepsilon)$.
- f je **polinomnog rasta**, ako f raste sličnom brzinom kao i neka pozitivna potencija, tj. $\exists a_1,\ a_2 > 0$ takvi da je $f(x) \in \Omega(x^{a_1})$ i $f(x) \in \mathcal{O}(x^{a_2})$.
- f je **blagog eksponencijalnog rasta**, ako f raste brže od bilo koje potencije, ali sporije od bilo koje eksponencijalne funkcije c^x, za $c > 1$. To znači da je $\forall a > 0,\ f(x) \in \Omega(x^a)$ i $\forall \varepsilon > 0,\ f(x) \in o((1 + \varepsilon^x))$.

Uvod u složenost

- f je **eksponencijalnog rasta**, ako postoje $c_1,\ c_2 > 1$ takvi da je $f(x) \in \Omega(c_1^x)$ i $f(x) \in \mathcal{O}(c_2^x)$.
- f je **nadeksponencijalnog rasta**, ako f raste brže od bilo koje eksponencijalne funkcije, tj. $\forall c > 1$ je $f(x) \in \Omega(c^x)$.

Netrivijalni primjeri klasifikacije u skladu s definicijom:

- $f(x) = logx$ je blagog rasta.
- $f(x) = x \cdot logx$ je polinomnog rasta.
- $f(x) = x^{logx}$ je blagog eksponencijalnog rasta.
- $f(x) = x \cdot 2^x$ je eksponencijalnog rasta.
- $f(x) = x^x$ je nadeksponencijalnog rasta.

Zadatak:

Zadano je cjelobrojno polje. Opišite algoritam za pronalaženje:

- podniza uzastopnih elemenata maksimalne sume
- rastućeg podniza (ne nužno uzastopnih elemenata) maksimalne duljine.

Rezultat izvršavanja je maksimalna suma, odnosno maksimalna duljina podniza.